目录

动态开点线段树

大意：给你一个长度 n 的数组，和两个数 l,r，问你有多少区间满足$l \le \sum_{i=l}^{r} a_i \le r$，输出即可。 思路：枚举区间右端点 j，即我们要找的就是有多少个 i，使得$i < j$且$l \le sum[j] - sum[i] \le r$，那么我们建一个线段树维护前缀和出现的次数，由于空间限制，我们只能动态开点，每次开点最坏是 log 级别的空间消耗，

```
t[++cnt]=0;
ls[cnt]=rs[cnt]=0;
x=cnt;
```

然后 剩下的就跟普通的线段树没什么区别拉

细节见代码：

```
#include<bits/stdc++.h>

#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define SZ(x)  (int)x.size()
#define all(x) x.begin(),x.end()

using namespace std;

const int N = 1e5 + 3;
const LL mod = 1e9 + 7;
const LL inf=1e10;
int n,a[N],l,r;
LL t[N*100];
int cnt;
LL ls[N*100],rs[N*100];
LL root;
void add(LL &x,LL l,LL r,LL d){
    if(!x){
        t[++cnt]=0;
        ls[cnt]=rs[cnt]=0;
        x=cnt;
    }
    if(l==r){
        t[x]++;
        return ;
    }
    LL mid=l+r>>1;
    if(d<=mid)add(ls[x],l,mid,d);
    else add(rs[x],mid+1,r,d);
    t[x]=t[ls[x]]+t[rs[x]];
}
LL get(int now,LL l,LL r,LL x,LL y){
    if(l>=x&&r<=y){
        return t[now];
    }
    LL ans=0;
    LL mid=l+r>>1;
    if(x<=mid&&ls[now])ans+=get(ls[now],l,mid,x,y);
    if(y>mid&&rs[now])ans+=get(rs[now],mid+1,r,x,y);
    return ans;
}
int main() {
```

```
    ios::sync_with_stdio(false);
    cin>>n>>l>>r;
    for(int i=1;i<=n;i++)cin>>a[i];
    add(root,-inf,inf,0);
    LL ans=0,sum=0;
    for(int i=1;i<=n;i++){
        sum+=a[i];
        ans+=get(root,-inf,inf,sum-r,sum-l);
        add(root,-inf,inf,sum);
    }
    cout<<ans<<endl;
    return 0;
}
```

cdq 分治模板(三维偏序)

```
#include<bits/stdc++.h>

#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define SZ(x)  (int)x.size()
#define all(x) x.begin(),x.end()

using namespace std;
const int N = 5e5 + 3;
const LL mod = 1e9 + 7;
int n,k,t[N],tot;
struct uzi{
    int a,b,c,tot,ans;
}p[N],q[N];
int ans[N];
int cmp1(uzi A,uzi B){
    if(A.a==B.a&&A.b==B.b)return A.c<B.c;
    if(A.a==B.a)return A.b<B.b;
    return A.a<B.a;
}
int cmp2(uzi A,uzi B){
    if(A.b==B.b)return A.c<B.c;
    return A.b<B.b;
}
void add(int p,int d){
    for(;p<=k;p+=p&-p)t[p]+=d;
}
int get(int p){
    int ans=0;
```

```
    for(;p;p-=p&-p)ans+=t[p];
    return ans;
}
void solve(int l,int r){
    if(l>=r)return ;
    int mid=(l+r)>>1;
    solve(l,mid);
    solve(mid+1,r);
    sort(q+l,q+1+mid,cmp2);
    sort(q+mid+1,q+1+r,cmp2);
    int j=l;
    for(int i=mid+1;i<=r;i++){
        while(j<=mid&&q[i].b>=q[j].b){add(q[j].c,q[j].tot);j++;}
        q[i].ans+=get(q[i].c);
    }
    for(int k=l;k<j;k++)add(q[k].c,-q[k].tot);
}
bool operator == (uzi A,uzi B){
    return A.a==B.a&&A.b==B.b&&A.c==B.c;
}
int main() {
    ios::sync_with_stdio(false);
    cin>>n>>k;
    for(int i=1;i<=n;i++)cin>>p[i].a>>p[i].b>>p[i].c;
    int l=1;
    sort(p+1,p+1+n,cmp1);
    while(l<=n){
        int pre=l;
        while(l+1<=n&&p[l]==p[l+1])l++;
        q[++tot]=p[l];
        q[tot].ans=0;
        q[tot].tot=l-pre+1;
        l++;
    }
    solve(1,tot);
    for(int i=1;i<=tot;i++)ans[q[i].ans+q[i].tot-1]+=q[i].tot;
    for(int i=0;i<n;i++)cout<<ans[i]<<endl;
    return 0;
}
```

rmq 模板

```
int dp[N + 33][21];
void RMQ() {
for (int i = 1; i <= n; i++) dp[i][0] = i;
for (int i = 1; (1 << i) <= N; i++) {
    for (int j = 1; j + (1 << i) - 1 <= N; j++) {
```

```
        dp[j][i] = a[dp[j][i - 1]] >= a[dp[j + (1 << (i - 1))][i - 1]] ?
                   dp[j][i - 1] : dp[j + (1 << (i - 1))][i - 1];
    }
}
}
int lg[N + 3];
void init() {
    lg[0] = -1;
    for (int i = 1; i < N; i++)lg[i] = lg[i >> 1] + 1;
}
int get(int L, int R) {
    int k = lg[R - L + 1];
    int Mid = (a[dp[L][k]] >= a[dp[R - (1 << k) + 1][k]] ? dp[L][k] : d
p[R - (1 << k) + 1][k]); //最大值下标
}
```

树套树模板

题目链接

树套树模板

```
#include<bits/stdc++.h>

#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define SZ(x)  (int)x.size()
#define all(x) x.begin(),x.end()

using namespace std;

const int N = 1e5 + 3;
const LL mod = 1e9 + 7;
int n,m,a[N],b[N<<1],cnT,tot,now,T[N<<1],Q[N*170],cnt;
struct uzi{int sta;char x;int l,r,k;}p[N];
int newnode(){int p=tot?Q[tot--]:++cnT;return p;}
struct faker{int sum,l,r;}t[N*170];
void up(int &x,int y,int l,int r,int pos,int val){
    if(!x){x=newnode();t[x].sum=t[x].l=t[x].r=0;}
    t[x]=t[y];t[x].sum+=val;
    if(l==r)return ;int mid=l+r>>1;
    if(pos<=mid)up(t[x].l,t[y].l,l,mid,pos,val);
    else up(t[x].r,t[y].r,mid+1,r,pos,val);
    if(!t[x].sum)Q[++tot]=x,x=0;
```

```
}
int totx,toty;
int L[100],R[100];
int get(int l,int r,int x){
    if(l==r)return l;
    int Y=0;
    int mid=l+r>>1;
    for(int i=1;i<=toty;i++)Y+=t[t[R[i]].l].sum;
    for(int i=1;i<=totx;i++)Y-=t[t[L[i]].l].sum;
    if(Y>=x){
        for(int i=1;i<=toty;i++)R[i]=t[R[i]].l;
        for(int i=1;i<=totx;i++)L[i]=t[L[i]].l;
        return get(l,mid,x);
    }
    else{
        for(int i=1;i<=toty;i++)R[i]=t[R[i]].r;
        for(int i=1;i<=totx;i++)L[i]=t[L[i]].r;
        return get(mid+1,r,x-Y);
    }
}
int main() {
    ios::sync_with_stdio(false);
    cin>>n>>m;
    for(int i=1;i<=n;i++)cin>>a[i],b[++cnt]=a[i];
    for(int i=1;i<=m;i++){
        int l,r,k;
        char x;
        cin>>x;
        if(x=='Q'){
            cin>>l>>r>>k;
            p[i]={0,x,l,r,k};
        }else {
            cin>>l>>r;
            p[i]={1,x,l,r,0};
            b[++cnt]=r;
        }
    }
    sort(b+1,b+1+cnt);
    cnt=unique(b+1,b+1+cnt)-b-1;
    for(int i=1;i<=n;i++)a[i]=lower_bound(b+1,b+1+cnt,a[i])-b;
    for(int i=1;i<=m;i++)if(p[i].sta)p[i].r=lower_bound(b+1,b+1+cnt,p
[i].r)-b;
    for(int i=1;i<=n;i++)for(int j=i;j<=n;j+=j&-j)up(T[j],T[j],1,cnt,a
[i],1);
    for(int i=1;i<=m;i++){
        if(!p[i].sta){
            totx=toty=0;
            for(int j=p[i].l-1;j;j-=j&-j)L[++totx]=T[j];
            for(int j=p[i].r;  j;j-=j&-j)R[++toty]=T[j];
```

```
            //cout<<get(1,cnt,p[i].k)<<' ';
            cout<<b[get(1,cnt,p[i].k)]<<endl;
        }else{
            for(int j=p[i].l;j<=n;j+=j&-j)up(T[j],T[j],1,cnt,a[p[i].l],
-1);
            a[p[i].l]=p[i].r;
            for(int j=p[i].l;j<=n;j+=j&-j)up(T[j],T[j],1,cnt,p[i].r,1);
        }
    }
    return 0;
}
```

大数

```
const int MAXN=150;
struct bign
{
    int len, s[MAXN];
    bign ()
    {
        memset(s, 0, sizeof(s));
        len = 1;
    }
    bign (int num) { *this = num; }
    bign (const char *num) { *this = num; }
    bign operator = (const int num)
    {
        char s[MAXN];
        sprintf(s, "%d", num);
        *this = s;
        return *this;
    }
    bign operator = (const char *num)
    {
        for(int i = 0; num[i] == '0'; num++) ;  //去前导0
        len = strlen(num);
        for(int i = 0; i < len; i++) s[i] = num[len-i-1] - '0';
        return *this;
    }
    bign operator + (const bign &b) const //+
    {
        bign c;
        c.len = 0;
        for(int i = 0, g = 0; g || i < max(len, b.len); i++)
        {
            int x = g;
            if(i < len) x += s[i];
            if(i < b.len) x += b.s[i];
            c.s[c.len++] = x % 10;
```

```
        g = x / 10;
    }
    return c;
}
bign operator += (const bign &b)
{
    *this = *this + b;
    return *this;
}
void clean()
{
    while(len > 1 && !s[len-1]) len--;
}
bign operator * (const bign &b) //*
{
    bign c;
    c.len = len + b.len;
    for(int i = 0; i < len; i++)
    {
        for(int j = 0; j < b.len; j++)
        {
            c.s[i+j] += s[i] * b.s[j];
        }
    }
    for(int i = 0; i < c.len; i++)
    {
        c.s[i+1] += c.s[i]/10;
        c.s[i] %= 10;
    }
    c.clean();
    return c;
}
bign operator *= (const bign &b)
{
    *this = *this * b;
    return *this;
}
bign operator - (const bign &b)
{
    bign c;
    c.len = 0;
    for(int i = 0, g = 0; i < len; i++)
    {
        int x = s[i] - g;
        if(i < b.len) x -= b.s[i];
        if(x >= 0) g = 0;
        else
        {
            g = 1;
```

```
            x += 10;
        }
        c.s[c.len++] = x;
    }
    c.clean();
    return c;
}
bign operator -= (const bign &b)
{
    *this = *this - b;
    return *this;
}
bign operator / (const bign &b)
{
    bign c, f = 0;
    for(int i = len-1; i >= 0; i--)
    {
        f = f*10;
        f.s[0] = s[i];
        while(f >= b)
        {
            f -= b;
            c.s[i]++;
        }
    }
    c.len = len;
    c.clean();
    return c;
}
bign operator /= (const bign &b)
{
    *this  = *this / b;
    return *this;
}
bign operator % (const bign &b)
{
    bign r = *this / b;
    r = *this - r*b;
    return r;
}
bign operator %= (const bign &b)
{
    *this = *this % b;
    return *this;
}
bool operator < (const bign &b)
{
    if(len != b.len) return len < b.len;
    for(int i = len-1; i >= 0; i--)
```

```
        {
            if(s[i] != b.s[i]) return s[i] < b.s[i];
        }
        return false;
    }
    bool operator > (const bign &b)
    {
        if(len != b.len) return len > b.len;
        for(int i = len-1; i >= 0; i--)
        {
            if(s[i] != b.s[i]) return s[i] > b.s[i];
        }
        return false;
    }
    bool operator == (const bign &b)
    {
        return !(*this > b) && !(*this < b);
    }
    bool operator != (const bign &b)
    {
        return !(*this == b);
    }
    bool operator <= (const bign &b)
    {
        return *this < b || *this == b;
    }
    bool operator >= (const bign &b)
    {
        return *this > b || *this == b;
    }
    string str() const
    {
        string res = "";
        for(int i = 0; i < len; i++) res = char(s[i]+'0') + res;
        return res;
    }
};
istream& operator >> (istream &in, bign &x)
{
    string s;
    in >> s;
    x = s.c_str();
    return in;
}
ostream& operator << (ostream &out, const bign &x)
{
    if (x.str()=="") out<<0;
    else out << x.str();
```

```
    return out;
}
```

拉格朗日插值

$$f(i)=\sum_{i=1}^{n}y_i\prod_{i\neq j}\frac{k-x_j}{x_i-x_j}$$

n 个点唯一确定一个 n-1 次的多项式

注意可以预处理阶乘优化

题目背景

这是一道模板题

题目描述

由小学知识可知，n 个点(x,y)可以唯一地确定一个多项式

现在，给定 n 个点，请你确定这个多项式，并将 k 代入求值

求出的值对 998244353 取模

```
#include<bits/stdc++.h>

#define LL long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back

using namespace std;

const int N = 2e5 +11;
const LL mod=998244353;
int n,k,a[N],b[N];
LL ans=0;
int main(){
    ios::sync_with_stdio(false);
    cin>>n>>k;
    for(int i=1;i<=n;i++)cin>>a[i]>>b[i];
    for(int i=1;i<=n;i++){
        LL res=b[i],L=1,R=1;
        for(int j=1;j<=n;j++){
            if(j==i)continue;
            L*=(k-a[j]+mod)%mod;
```

```
            L%=mod;
            R*=(a[i]-a[j]+mod)%mod;
            R%=mod;
        }
        L*=powmod(R,mod-2,mod);
        L%=mod;
        ans+=(res*L)%mod;
        ans%=mod;
    }
    cout<<ans<<endl;

    return 0;
}
```

数列分块

给出一个长为 n 的数列，以及 n 个操作，操作涉及区间询问等于一个数 c 的元素，并将这个区间的所有元素改为 c

```
#include<bits/stdc++.h>

#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define SZ(x)  (int)x.size()
#define all(x) x.begin(),x.end()

using namespace std;

const int N = 5e5 + 3;
const LL mod = 1e9 + 7;
const int INF=INT32_MIN;
int n,a[N],x,tmp[N];
int get(int  t){
    return t/x+(t%x?1:0);
}
int main() {
    ios::sync_with_stdio(false);
    cin>>n;x=sqrt(n);
    for(int i=1;i<=n;i++)cin>>a[i];
    for(int i=get(1);i<=get(n);i++)tmp[i]=INF;
    for(int i=1;i<=n;i++){
        int l,r,c;cin>>l>>r>>c;
        int L=get(l);
        int res=0;
```

```
        for(;l<=L*x&&l<=r;){
            if(tmp[L]!=INF){
                for(int j=(L-1)*x+1;j<=L*x;j++)a[j]=tmp[L];
                tmp[L]=INF;
            }
            if(a[l]!=c)a[l]=c;
            else res++;
            if(l<L*x)l++;
            else break;
        }
        int R=get(r);
        for(;r>l&&r>=(R-1)*x+1;){
            if(tmp[R]!=INF){
                for(int j=(R-1)*x+1;j<=R*x;j++)a[j]=tmp[R];
                tmp[R]=INF;
            }
            if(a[r]!=c)a[r]=c;
            else res++;

            if(r>(R-1)*x+1)r--;
            else break;
        }
        l++;r--;
        for(;l<=r;l+=x){
            if(tmp[get(l)]==INF){
                tmp[get(l)]=c;
                for(int j=(get(l)-1)*x+1;j<=(get(l)*x);j++){
                    if(a[j]==c)res++;
                }
            }else{
                if(tmp[get(l)]!=c)tmp[get(l)]=c;
                else res+=x;
            }
        }
        cout<<res<<endl;
    }
    return 0;
}
```

整体二分

给你一个矩阵询问子矩阵的第 k 小

整体二分练习题,就是多了一个二维前缀和,直接二维树状数组就行了

```
#include<bits/stdc++.h>

#define fi first
```

```
#define se second
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define SZ(x)  (int)x.size()
#define all(x) x.begin(),x.end()

using namespace std;

const int N = 5e5 + 3;
const LL mod = 1e9 + 7;
int n,q;
int a[555][555];
struct uzi{
    LL o,x,y,a,b,d,pos;
}p[N],q1[N],q2[N];
int cnt;
LL ans[N];
LL t[555][555];
void add(int k,int p,LL d){
    for(int i=k;i<=n;i+=i&-i){
        for(int j=p;j<=n;j+=j&-j)t[i][j]+=d;
    }
}
LL get(int a,int b){
    LL A=0;
    if(!a||!b)return 0;
    for(int i=a;i;i-=i&-i){
        for(int j=b;j;j-=j&-j)A+=t[i][j];
    }
    return A;
}
void solve(LL l,LL r,int x,int y){
    if(l>r||x>y)return ;
    if(l==r){
        for(int i=x;i<=y;i++)if(p[i].o==2)ans[p[i].pos]=l;
        return ;
    }
    int c=0,d=0;
    LL mid=l+r>>1;
    for(int i=x;i<=y;i++){
        if(p[i].o==1){
            if(p[i].a<=mid)add(p[i].x,p[i].y,1),q1[++c]=p[i];
            else q2[++d]=p[i];
        }else{
            LL res=get(p[i].a,p[i].b)-get(p[i].a,p[i].y-1)-get(p[i].x-1,
p[i].b)+get(p[i].x-1,p[i].y-1);
            if(res>=p[i].d)q1[++c]=p[i];
```

```
            else p[i].d-=res,q2[++d]=p[i];
        }
    }
    for(int i=1;i<=c;i++)if(q1[i].o==1)add(q1[i].x,q1[i].y,-1);
    for(int i=x;i<=x+c-1;i++)p[i]=q1[i-x+1];
    for(int i=x+c;i<=y;i++)p[i]=q2[i-x-c+1];
    solve(l,mid,x,x+c-1);
    solve(mid+1,r,x+c,y);
}
int main() {
    ios::sync_with_stdio(false);
    cin>>n>>q;
    for(int i=1;i<=n;i++)for(int j=1;j<=n;j++)cin>>a[i][j],p[++cnt]={1,
i,j,a[i][j],0,0,0};
    for(int i=1;i<=q;i++){
        int x1,x2,y1,y2,k;
        cin>>x1>>y1>>x2>>y2>>k;
        p[++cnt]={2,x1,y1,x2,y2,k,i};
    }
    solve(-1e10,1e10,1,cnt);
    for(int i=1;i<=q;i++)cout<<ans[i]<<endl;
    return 0;
}
```

树套树

题意：给你两个长度为 n,m 的排列 a,b，然后让你支持两个操作： 1.询问 a 排列上[l1,r1]区间和 b 排列上[l2,r2]区间相同元素的个数。 2.交换 b 排列上 c,d 位置的元素

思路 ：我们可以建立一个长度 m 的数组 c，其中第i个元素表示 b[i]在 a 中的位置，那么显然答案就是 c 数组[l2,r2]中值在[l1,r1]的个数了。 直接上树状数组套主席树即可在线解决这个问题，注意要回收空间。

```
#include<bits/stdc++.h>

#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define SZ(x)  (int)x.size()
#define all(x) x.begin(),x.end()

using namespace std;
```

```
const int N = 2e5 + 3;
const LL mod = 1e9 + 7;
int n,m,a[N],b[N],A[N],B[N],T[N],tot,Q[N*40],cnt;
struct uzi{int sum,l,r;}t[N*170];
int newnode(){int p=tot?Q[tot--]:++cnt;t[p].sum=t[p].l=t[p].r=0;return
p;}
void up(int &x,int y,int l,int r,int p,int v){
    if(!x){x=newnode();}t[x].sum+=v;
    if(l==r)return ;int mid=l+r>>1;
    if(p<=mid)up(t[x].l,t[y].l,l,mid,p,v);
    else up(t[x].r,t[y].r,mid+1,r,p,v);
    if(!t[x].sum)Q[++tot]=x,x=0;
}
int get(int o,int l,int r,int x,int y){
    if(x<=l&&y>=r)return t[o].sum;
    int mid=l+r>>1,ans=0;
    if(x<=mid)ans+=get(t[o].l,l,mid,x,y);
    if(y>mid)ans+=get(t[o].r,mid+1,r,x,y);
    return ans;
}
int main() {
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)scanf("%d",&a[i]),A[a[i]]=i;
    for(int i=1;i<=n;i++)scanf("%d",&b[i]),B[b[i]]=i;
    for(int i=1;i<=n;i++)for(int j=i;j<=n;j+=j&-j)up(T[j],T[j],1,n,B[a
[i]],1);
    for(int i=1;i<=m;i++){
        int o,la,ra,lb,rb;
        cin>>o;
        if(o==1){
            scanf("%d%d%d%d",&la,&ra,&lb,&rb);
            int ANS=0;
            for(int j=la-1;j;j-=j&-j)ANS-=get(T[j],1,n,lb,rb);
            for(int j=ra;j;j-=j&-j)ANS+=get(T[j],1,n,lb,rb);
            printf("%d\n",ANS);
        }else{
            scanf("%d%d",&la,&lb);
            int pA=A[b[la]],pB=A[b[lb]];
            for(int j=pA;j<=n;j+=j&-j)up(T[j],T[j],1,n,B[b[la]],-1);
            for(int j=pB;j<=n;j+=j&-j)up(T[j],T[j],1,n,B[b[lb]],-1);
            swap(b[la],b[lb]);
            B[b[la]]=la;
            B[b[lb]]=lb;
            pA=A[b[la]];
            pB=A[b[lb]];
            for(int j=pA;j<=n;j+=j&-j)up(T[j],T[j],1,n,B[b[la]],1);
            for(int j=pB;j<=n;j+=j&-j)up(T[j],T[j],1,n,B[b[lb]],1);
        }
```

```
    }
    return 0;
}
```

树连剖分

很早就想学的模板，由于懒拖到现在（其实是菜） 树链剖分其实是将树形结构处理成线性序列然后用数据结构来维护树的一个东西。可以解决很多树上问题。 具体重要的三个函数为： 一.dfs1 第一遍 dfs 遍历这颗树，处理出每个点的子树大小（包含自己）为siz，节点的父亲节点为fa，节点的深度为de,还有每个节点的重儿子(siz 值最大的儿子)

```
void dfs1(int now,int pre,int d){
    siz[now]=1;
    fa[now]=pre;
    de[now]=d;
    int cnt=-1;
    for(auto k:v[now]){
        if(k==pre)continue;
        dfs1(k,now,d+1);
        siz[now]+=siz[k];
        if(siz[k]>cnt){
            cnt=siz[k];
            son[now]=k;
        }
    }
}
```

二.dfs2 处理出每个节点所在链的顶节点记为to，和该节点所在 dfs 序列的标号，和 dfs 序列的每个元素的值，每个节点先处理重儿子节点.

```
void dfs2(int now,int pre){
    to[now]=pre;
    a[now]=++cnt;
    t[cnt]=x[now];
    t[cnt]%=p;
    if(!son[now])return ;
    dfs2(son[now],pre);
    for(auto k:v[now]){
        if(k==fa[now]||k==son[now])continue;
        dfs2(k,k);
    }
}
```

三.找最近公共祖先 对两个点，当他们不在一条链上时，我们对他们所在链顶端深度较大的那个点就行跳跃，跳到顶的父亲节点，不断重复。 然后返回深度较小的点即为 lca。

```
int find_lca(int l,int r){
    int res=0;
    while(to[l]!=to[r]){
        if(de[to[l]]<de[to[r]])swap(l,r);
        l=fa[to[l]];
    }
    if(de[l]>de[r])swap(l,r);
    return l;
}
```

对洛谷那题来说就是加上了线段树来维护区间的一些信息，重要的还是树链剖分的部分

下面是 ac 的代码

```
#include<bits/stdc++.h>

#define LL long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back

using namespace std;

const int N = 1e5 +11;
int siz[N],fa[N],son[N],a[N],cnt,to[N],t[N],de[N];
int T[N<<2];
int laz[N<<2];
vector<int>v[N];
int n,m,p,r,x[N];
void dfs1(int now,int pre,int d){
    siz[now]=1;
    fa[now]=pre;
    de[now]=d;
    int cnt=-1;
    for(auto k:v[now]){
        if(k==pre)continue;
        dfs1(k,now,d+1);
        siz[now]+=siz[k];
        if(siz[k]>cnt){
            cnt=siz[k];
            son[now]=k;
        }
    }
}
void dfs2(int now,int pre){
    to[now]=pre;
    a[now]=++cnt;
```

```
    t[cnt]=x[now];
    t[cnt]%=p;
    if(!son[now])return ;
    dfs2(son[now],pre);
    for(auto k:v[now]){
        if(k==fa[now]||k==son[now])continue;
        dfs2(k,k);
    }
}

void build(int now,int l,int r){
    if(l==r){
        T[now]=t[l];
        return ;
    }
    int mid=l+r>>1;
    build(now<<1,l,mid);
    build(now<<1|1,mid+1,r);
    T[now]=T[now<<1]+T[now<<1|1];
    T[now]%=p;
}
void pd(int now,int l,int r){
    if(laz[now]){
        int mid=l+r>>1;
        laz[now<<1]+=laz[now];
        laz[now<<1|1]+=laz[now];
        T[now<<1]+=(1ll*laz[now]*(mid-l+1))%p;
        T[now<<1|1]+=(1ll*laz[now]*(r-mid))%p;
        T[now<<1]%=p;
        T[now<<1|1]%=p;
        laz[now<<1]%=p;
        laz[now<<1|1]%=p;
        laz[now]=0;
    }
}
int get(int now,int l,int r,int x,int y){
    if(l>=x&&r<=y){
        return (T[now])%p;
    }
    pd(now,l,r);
    int mid=l+r>>1;
    int res=0;
    if(x<=mid)res+=get(now<<1,l,mid,x,y);
    if(y>mid)res+=get(now<<1|1,mid+1,r,x,y);
    res%=p;
    return res;
}
void ud(int now,int l,int r,int x,int y,int d){
    if(l>=x&&r<=y){
```

```
        laz[now]+=d;
        laz[now]%=p;
        T[now]+=(1ll*d*(r-l+1)%p);
        T[now]%=p;
        return;
    }
    pd(now,l,r);
    int mid=l+r>>1;
    if(x<=mid)ud(now<<1,l,mid,x,y,d);
    if(y>mid)ud(now<<1|1,mid+1,r,x,y,d);
    T[now]=T[now<<1]+T[now<<1|1];
    T[now]%=p;
}
int gao(int l,int r){
    int res=0;
    while(to[l]!=to[r]){
        if(de[to[l]]<de[to[r]])swap(l,r);
        res+=get(1,1,cnt,a[to[l]],a[l]);
        res%=p;
        l=fa[to[l]];
    }
    if(de[l]>de[r])swap(l,r);
    res+=get(1,1,cnt,a[l],a[r]);
    return res%p;
}
void gan(int l,int r,LL k){
    k%=p;
    while(to[l]!=to[r]){
        if(de[to[l]]<de[to[r]])swap(l,r);
        ud(1,1,cnt,a[to[l]],a[l],k);
        l=fa[to[l]];
    }
    if(de[l]>de[r])swap(l,r);
    ud(1,1,cnt,a[l],a[r],k);
    return ;
}
int main(){
    ios::sync_with_stdio(false);
    cin>>n>>m>>r>>p;
    for(int i=1;i<=n;i++)cin>>x[i];
    for(int i=1;i<n;i++){
        int s,t;
        cin>>s>>t;
        v[s].pb(t);
        v[t].pb(s);
    }
    dfs1(r,0,0);
    dfs2(r,r);
    cout<<endl;
```

```
    build(1,1,cnt);
    for(int i=1;i<=m;i++){
        int ope;
        int X,Y,Z;
        cin>>ope;
        if(ope==1){
            cin>>X>>Y>>Z;
            gan(X,Y,Z);
        }else if(ope==2){
            cin>>X>>Y;
            cout<<gao(X,Y)<<endl;
        }else if(ope==3){
            cin>>X>>Z;
            ud(1,1,cnt,a[X],a[X]+siz[X]-1,Z);
        }else{
            cin>>X;
            cout<<get(1,1,cnt,a[X],a[X]+siz[X]-1)<<endl;
        }
    }
    return 0;
}
```

点分治

每次找重心，然后处理每个子树。

```
// luogu-judger-enable-o2
#include<bits/stdc++.h>

#define fi first
#define se second
#define pb push_back
#define mp make_pair
#define LL long long
#define SZ(X) X.size()
#define pii pair<int,int>
#define ALL(X) X.begin(),X.end()

using namespace std;

const int N = 1e4 + 11;
int n, m;
vector<pii>v[N];
int q[N];
int rt, son[N], sz[N], vis[N];
void root(int now, int pre) { //找重心
    sz[now] = 1;
    son[now] = 0;
```

```
    for (auto k : v[now]) {
        if (vis[k.fi] || k.fi == pre)continue;
        root(k.fi, now);
        sz[now] += sz[k.fi];
        son[now] = max(son[now], sz[k.fi]);
    }
    son[now] = max(son[now], n - sz[now]);
    if (!rt || son[rt] > son[now]) {
        rt = now;
    }
    return ;
}
int a[N], dis[10000003];
int md[N], MK, cnt;
void getdis(int now, int pre, int di) {
    if (di > MK)return ;
    md[++cnt] = di;
    for (auto k : v[now]) {
        if (vis[k.fi] || k.fi == pre)continue;
        getdis(k.fi, now, di + k.se);
    }
}
int qa[N], tmp;
void get(int now, int pre) {
    dis[0] = 1;
    for (auto k : v[now]) {
        if (k.fi == pre || vis[k.fi])continue;
        cnt = 0;
        getdis(k.fi, now, k.se);
        for (int i = 1; i <= cnt; ++i) {
            for (int j = 1; j <= m; j++) {
                if (q[j] >= md[i])a[j] |= dis[q[j] - md[i]];
            }
        }
        for (int i = 1; i <= cnt; i++)dis[md[i]] = 1, qa[++tmp] = md[i];
    }
}
void dfs(int now) {
    vis[now] = 1;
    tmp = 0;
    get(now, 0);
    for (int i = 1; i <= tmp; i++)dis[qa[i]] = 0;
    for (auto k : v[now]) {
        if (vis[k.fi])continue;
        rt = 0;
        n = sz[k.fi];
        root(k.fi, 0);
        dfs(rt);
    }
```

```
    return ;
}
int main() {
    // freopen("1.in", "r", stdin);
    scanf("%d %d", &n, &m);
    for (int i = 1; i < n; i++) {
        int s, t, w;
        scanf("%d%d%d", &s, &t, &w);
        v[s].pb({t, w});
        v[t].pb({s, w});
    }
    for (int i = 1; i <= m; i++)scanf("%d", &q[i]), MK = max(q[i], MK);
    root(1, 0);
    root(rt, 0);
    dfs(rt);
    for (int i = 1; i <= m; i++)puts(a[i] ? "AYE" : "NAY");
    return 0;
}
```

蔡勒公式（日期->星期）

$w=\left(y+\left[\frac{y}{4}\right]+\left[\frac{c}{4}\right]-2c+\left[\frac{26(m+1)}{10}\right]+d-1\right)mod7$ 公式中的符号含义如下：

w：星期（计算所得的数值对应的星期：0-星期日；1-星期一；2-星期二；3-星期三；4-星期四；5-星期五；6-星期六）[注 1] c：年份前两位数 y：年份后两位数 m：月（m 的取值范围为 3 至 14，即在蔡勒公式中，某年的 1、2 月要看作上一年的 13、14 月来计算，比如 2003 年 1 月 1 日要看作 2002 年的 13 月 1 日来计算） d：日 [ ]：称作高斯符号，代表向下取整，即，取不大于原数的最大整数。

矩形面积并

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <algorithm>
#include <cstdio>
using namespace std;
typedef long long ll;
#define wfor(i,j,k) for(i=j;i<k;++i)
#define mfor(i,j,k) for(i=j;i>=k;--i)
// void read(int &x) {
//  char ch = getchar(); x = 0;
//  for (; ch < '0' || ch > '9'; ch = getchar());
//  for (; ch >= '0' && ch <= '9'; ch = getchar()) x = x * 10 + ch - '0
';
// }
```

```
const int maxn = 2005;
struct node
{
    double len;
    int s;
    int l;
    int r;
};
node tree[maxn << 2];
struct Line
{
    int l;
    int r;
    int high;
    int flag;
    bool operator <(const Line other)const
    {
        return high < other.high;
    }
};
struct Point
{
    double x1;
    double y1;
    double x2;
    double y2;
};
Point point[maxn];
vector <double>num;
Line line[maxn << 2];
void push_up(int id)
{
    if (tree[id].s)
        tree[id].len = num[tree[id].r - 1] - num[tree[id].l - 1];
    else
        tree[id].len = tree[id << 1].len + tree[id << 1 | 1].len;
}
void build (int l, int r, int id)
{
    tree[id].len = 0;
    tree[id].s = 0;
    tree[id].l = l;
    tree[id].r = r;
    if (l + 1 == r)
    {
        return ;
    }
    int mid = (l + r) >> 1;
    build(l, mid, id << 1);
```

```
    build(mid, r, id << 1 | 1);
}
void update(int id, int L, int R, int number)
{
    if (tree[id].l >= L && tree[id].r <= R)
    {
        tree[id].s += number;
        push_up(id);
        return ;
    }
    int mid = (tree[id].l + tree[id].r) >> 1;
    if (mid > L)
        update(id << 1, L, R, number);
    if (mid < R)
        update(id << 1 | 1, L, R, number);
    push_up(id);
}
int main()
{
    std::ios::sync_with_stdio(false);
    int n;
    int casecnt = 0;
    while (cin >> n && n)
    {
        num.clear();
        casecnt++;
        int i;
        wfor(i, 0, n)
        {
            cin >> point[i].x1 >> point[i].y1 >> point[i].x2 >> point
[i].y2;
            num.push_back(point[i].x1);
            num.push_back(point[i].x2);
            num.push_back(point[i].y1);
            num.push_back(point[i].y2);
        }
        sort(num.begin(), num.end());
        auto it = unique(num.begin(), num.end());
        int len = it - num.begin();
        int cnt = 0;
        wfor(i, 0, n)
        {
            int x1, y1, x2, y2;
            x1 = lower_bound(num.begin(), it, point[i].x1) - num.begin()
 + 1;
            x2 = lower_bound(num.begin(), it, point[i].x2) - num.begin()
 + 1;
            y1 = lower_bound(num.begin(), it, point[i].y1) - num.begin()
 + 1;
```

```
            y2 = lower_bound(num.begin(), it, point[i].y2) - num.begin()
 + 1;
            line[cnt].l = x1;
            line[cnt].r = x2;
            line[cnt].flag = 1;
            line[cnt++].high = y1;
            line[cnt].l = x1;
            line[cnt].r = x2;
            line[cnt].flag = -1;
            line[cnt++].high = y2;
        }
        sort(line, line + cnt);
        build(1, len, 1);
        update(1, line[0].l, line[0].r, line[0].flag);
        int last = line[0].high;
        double ans = 0;
        wfor(i, 1, cnt)
        {
            ans += tree[1].len * (num[line[i].high - 1] - num[last -
1]);
            update(1, line[i].l, line[i].r, line[i].flag);
            last = line[i].high;
        }
        cout << "Test case #" << casecnt << endl;
        cout << "Total explored area: ";
        cout << fixed << setprecision(2) << ans << endl;
        cout << endl;
    }
    return 0;
}
```